Proceedings of the of the 17th Conference on Computer Science and Intelligence Systems pp. 205–209
DOI: 10.15439/2022F48
ISSN 2300-5963 ACSIS, Vol. 30

Insights into Neural Architectures for Learning Numerical Concepts from Simple Visual Data

Andrzej Śluzek
Warsaw University of Life Sciences-SGGW
ul. Nowoursynowska 166, 02-787 Warszawa, Poland
Email: andrzej_sluzek@sggw.edu.pl

***Abstract*—The paper reports some results on neural architectures for learning numerical concepts from visual data. We use datasets of small images with single-pixel dots (*one* to *six* per image) to learn the abstraction of small integers, and other numerical concepts (e.g. *even* versus *odd* numbers). Both fully-connected and convolutional architectures are investigated. The obtained results indicate that two categories of numerical properties apparently exist (in the context of discussed problems). In the first category, the properties can be learned without acquiring the counting skills, e.g. the notion of *small*, *medium* and *large* numbers. In the second category, explicit learning of counting is embedded into the architecture so that the concepts are learned from numbers rather than directly from visual data. In general, we find that CNN architectures (if properly crafted) are more efficient in the discussed problems and (additionally) come with more plausible explainability.**

I. Introduction and Motivation

THE CONCEPTS of numbers and numerical properties develop primarily (e.g. [1], [2]) from sensory experiences, with visual inspection playing the pivotal role.

Researchers investigate the topic of learning numerical abstractions mainly (apart from actual and prospective applications) as a challenging AI/ML problem.

Initially, the works were focused on object counting rather than understanding the abstractions, with efforts on *counting-by-localizing* sub-tasks, e.g. [3]. This was an application-oriented approach, and some sources (e.g. [4]) indicate that true understanding of numbers may not be even needed to perform counting tasks in visual data.

Later, researchers expressed more interests in grasping/learning the concept of numbers and numerical abstractions. First, it was done in the context of human brain functioning (e.g. [5]) but recent works focus on machine learning aspects as well. In particular, visual data have been explicitly used as inputs to learning algorithms and architectures, e.g. [6], [7], [8]. Complexity of those visual inputs is usually limited to avoid complicated image processing sub-tasks, i.e. either binary [6], [7] or near-binary [8] low-resolution images are used.

In this paper, we follow the same approach. Using results of [6] and [8] as the starting point, we attempt to develop simple neural architectures for learning the concept of numbers (from a limited range 1 to 6) and other numerical abstractions which can exist within such a narrow range of integers. Examples of such abstract concepts are:

- *even* and *odd* numbers;
- *small* $(1, 2)$, *medium* $(3, 4)$ and *large* $(5, 6)$ numbers;
- etc.

Formally, each concept is a division of integers (from 1 to 6 range) into classes. For example, enumeration from 1 to 6 corresponds to six classes (1), (2), (3), (4), (5) and (6), *even* and *odd* numbers form two classes $(1, 3, 5)$ and $(2, 4, 6)$, etc. Then, an input image should be assigned to the correct class based on the number of dots it contains.

In Section II, we explain the proposed methodology and overview the developed datasets. The considered neural architectures are also briefly explained. Section III presents the conducted experiments and achieved performances. Informal explanations of the trained architectures are provided there as well. The concluding Section IV highlights the most significant facts, underlines unsolved problems and proposes directions for the future work.

II. Methodology

A. Assumptions

In [6], two neural models are proposed for estimating numbers of white non-overlapping rectangles (up to 10) in small black images of 28×28 resolution.

In [7], more general (but still simple) tasks are discussed, i.e. estimating either the numbers of white isolated pixels or white connected components in 256×256 black images. The assumed numbers of counted objects are much larger (e.g. up to $3,000$ isolated pixels).

In [8], the task is to count isolated pixels (up to 10) in 10×10 images. However, the images are only approximately binary with random polarity (bright pixels on dark backgrounds or another way around).

In this paper, the proposed scenario (based on the above approaches) is further simplified. Images are very small (7×7) and contain up to 6 dots. Such small numbers of dots are motivated by a well known psychological fact that humans can visually perceive (without explicit counting) at most 7 objects. The images are only approximately binary and their polarity is random.

We consider both fully-connected (FcNN) and convolutional (CNN) neural networks. This is motivated by inconlusive reports from the past papers, where *pros* and *cons* of both

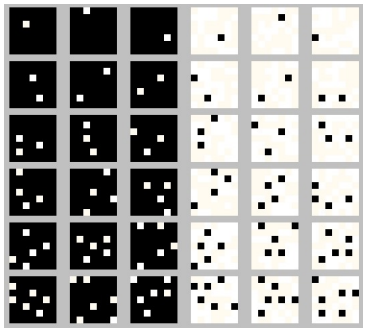

Fig. 1. Exemplary dataset images.

TABLE I
FIRST CNN ARCHITECTURE (CNN1).

Layer	Parameters	Activation
input	7×7	
conv.	$16 \times 3 \times 3$, str=1	relu
maxpool	2×2, str=1	
conv.	$16 \times 16 \times 2 \times 2$, str=1	relu
fc	10 outputs	tanh
fc	6 (or 2, or 3) outputs	softmax
output	6 (or 2, or 3)	

architectures are highlighted. Obviously, the considered architectures are rather simple to reflect small size and low complexity of input images.

B. Datasets

The developed dataset consists of $12,000$ 7×7 near-binary images ($6,000$ dark images with bright dots and $6,000$ images of the opposite polarity). The numbers of dots range from 1 to 6 (each number in $2,000$ images).

Near-binary images are used to alleviate overfitting, and to more realistically represent the real-world visual conditions.

Figure 1 shows a small sample of dataset images.

For the actual training and testing, the dataset is divided into two parts of $6,000$ images each (with the same ratio of all types of images). Only one half is used for training and validation, while the other half is used for testing only.

C. Neural architectures

As mentioned earlier, both FcNN and CNN architectures are considered. After extensive *try-and-error* tests (and partially following the ideas from [6] and [8]) we eventually propose two CNN and two FcNN architectures shown in Tables I- IV.

It can be noticed that (CNN1, CNN2) and (FcNN1, FcNN2) pairs are very similar. Actually, '*variant* 2' architectures are obtained by embedding '*variant* 1' (with 6 nodes in the terminal layer) and adding an additional FC layer. In the embedded '*variant* 1', its last layer is assumed to learn numbers from 1 to 6. Thus, the terminal layer of '*variant* 2' architectures would infer the numerical abstractions from presumably learned concepts of enumeration. This is further explained in Section III (with some special cases separately discussed).

TABLE II
SECOND CNN ARCHITECTURE (CNN2).

Layer	Parameters	Activation
input	7×7	
conv.	$16 \times 3 \times 3$, str=1	relu
maxpool	2×2, str=1	
conv.	$16 \times 16 \times 2 \times 2$, str=1	relu
fc	10 outputs	tanh
fc	6 outputs	softmax
fc	2 (or 3) outputs	softmax
output	2 (or 3)	

TABLE III
FIRST FCNN ARCHITECTURE (FCNN1)

Layer	Parameters	Activation
input	49	
fc	53 outputs	tanh
fc	10 outputs	tanh
fc	6 (or 2, or 3) outputs	softmax
output	6 (or 2, or 3)	

Actually, the architectures can be much slimmer than presented here. For example, the first hidden layer of FcNN architectures can be reduced from 53 nodes to, for example, 20 nodes. The current size is proposed to satisfy theoretical conditions of approximation theorems.

Similarly, the number of filters in convolutional layers of CNN architectures can be much smaller (see Section III); the proposed numbers just provide safe margins.

III. EXPERIMENTS AND RESULTS

The proposed architectures were separately trained for various concepts, including:

- learning the concept of numbers from 1 to 6;
- differentiating between *even* and *odd* numbers;
- differentiating between *medium* $(3, 4)$ numbers and other numbers (i.e. either *small* $(1, 2)$ or *large* $(5, 6)$);

TABLE IV
SECOND FCNN ARCHITECTURE (FCNN2)

Layer	Parameters	Activation
input	49	
fc	53 outputs	tanh
fc	10 outputs	tanh
fc	6 outputs	softmax
fc	2 (or 3) outputs	softmax
output	2 (or 3)	

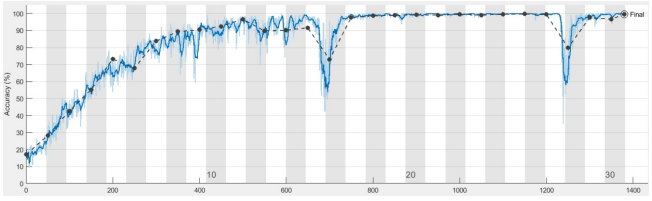

Fig. 2. Training (continuous line) and validation (circles) accuracy plots for CNN1 learning numbers from 1 to 6.

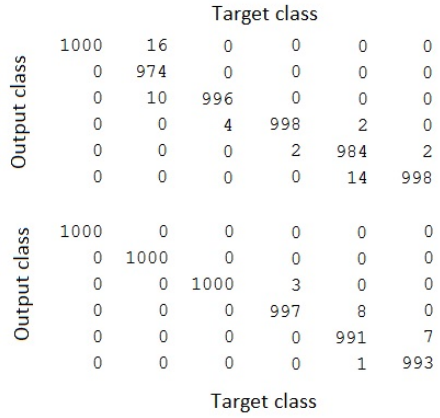

Fig. 3. Confusion matrices for FcNN1 (top) and for CNN1 (bottom) trained to identify numbers from 1 to 6.

Fig. 4. Examples of images with incorrectly identified numbers of dots by CNN1.

- and a few other (sometimes slightly artificial) numerical concepts.

In the following subsections, we discuss how the architectures can handle the above tasks.

A. Learning numbers from 1 to 6

This task is the only one similar to the problems discussed in earlier papers (e.g. [4], [6], [7], [8], [9]).

We found that both CNN1 and FcNN1 architectures can easily learn to nearly flawlessly recognize the number of dots in test dataset images. As an example, Fig. 2 shows training performances of CNN1.

The overall accuracies (on the test dataset, see Section II-B) are almost the same, i.e. 99.68% for CNN1 and 99.17% for FcNN1. However, the confusion matrices (given in Fig. 3) indicate more plausible performances of CNN1. Errors are located within classes with larger numbers of dots, i.e. in the scenarios where humans can make mistakes more frequently.

Examples of some incorrectly identified images are given in Fig. 4. At the first glance, they may look confusing even for humans.

B. Learning even and odd numbers

In this task, classification is not directly related to magnitudes of numbers but to their specific quantifiers (which should be identified by trained networks). In [5], it is argued that in human perception knowledge of numbers makes important contributions to acquiring meanings of such quantifiers. Our observations confirm this conclusion.

We find that 2-output architectures of CNN1 and FcNN1 type (i.e. the hidden layer for learning numbers is missing) are apparently unable to learn the abstraction of *even* and *odd* numbers. The most typical accuracy for CNN1 is the random-choice 50.00%. FcNN1 performs slightly better, reaching 60.48%. Thus, the architectures are unable to learn such numerical abstractions.

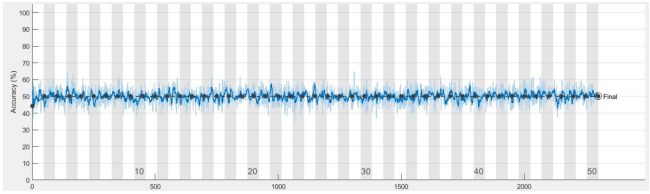

Fig. 5. Training (continuous line) and validation (circles) accuracy plots for CNN1 learning *even* and *odd* numbers.

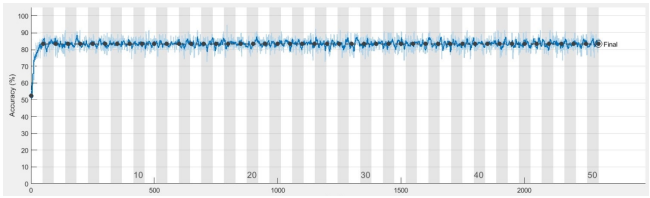

Fig. 6. Training (continuous line) and validation (circles) accuracy plots for CNN2 learning *even* and *odd* numbers.

An exemplary plot of CNN1 performances during training is shown in Fig. 5.

When '*variant* 2' architectures are used, the situation changes. Although accuracy of FcNN2 architecture deteriorates compared to FcNN1 (58.52% *versus* 60.48%), a significant improvement can be noticed for CNN2. Accuracy reaches 83.32% and training is very fast, as shown in Fig. 6.

In other words, if the concept of numerical values is embedded in the process (as explained in Section II-C) the convolutional architecture is able to quickly generalize the abstraction of *even* and *odd* numbers with reasonable accuracy. This can be considered a kind-of-projection of [5] observations onto machine learning domain.

C. Learning other numerical abstractions

The other exemplary numerical abstractions considered in the experiments are:

- *medium* numbers, i.e. $(3, 4)$ *versus* all other numbers,
- various classes of *compact* and *disconnected* subsets of integers (see Table V).

In the first experiment, the overall conclusions are similar to Section III-B. CNN1 architecture achieves results equivalent to random choice (66.67% accuracy; a training plot shown in Fig. 7) while FcNN can reach 74.36%.

Again, a switch from CNN1 to CNN2 architecture significantly improves performances. With the concept of numbers embedded, CNN2 achieves almost perfect 99.38% accuracy, and the learning curve is very steep (see Fig. 8). For FcNN2, however, there is no real improvement.

In other experiments, various classes have been proposed within the range $1, 6$; examples are shown in Table V. We

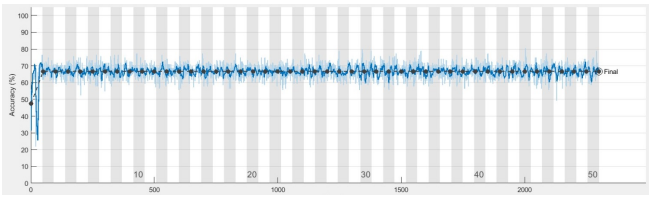

Fig. 7. Training (continuous line) and validation (circles) accuracy plots for CNN1 learning to distinguish $\{3, 4\}$ from other numbers.

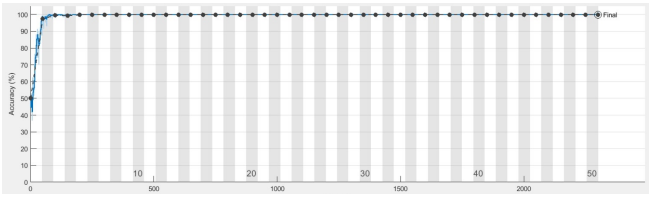

Fig. 8. Training (continuous line) and validation (circles) accuracy plots for CNN2 learning to distinguish $\{3, 4\}$ from other numbers.

found that learning abilities of proposed architectures strongly depend on whether the classes consist of compact subsets of integers (e.g. *even* and *odd* numbers are disconnected classes).

For FcNN architectures (regardless the variant) the results are satisfactory only if the classes are *compact*. Table V contains a number of examples with the top accuracies obtained by FcNN architectures (either FcNN1 or FcNN2). The accuracy is always very high for *compact* classes, while for *disconnected* classes the results are unacceptable (often even below the level of random choice).

For CNN architectures the results are generally much better. For *compact* classes, almost perfect accuracies can be obtained both by CNN1 and CNN2, while for *disconnected* classes only CNN2 are able to reach very high accuracies. CNN1 architectures usually struggle to go beyond the random-choice level.

However, some interesting examples of *disconnected* classes have been identified, where even CNN1 are able to score near-perfect performances. Learning numbers divisible by 3 (i.e. $(3, 6)$ and $(1, 2, 4, 5)$ classes) is one of such examples.

As expected, FcNN architectures perform rather poorly (63.25% for FcNN1 and 64.92% for FcNN2) in this example. However, CNN architectures can almost perfectly grasp this abstraction, even in CNN1 variant (i.e. without the embedded concept of numericals). Accuracy reaches 99.97% for the test

TABLE V
ACCURACY OF FCNN ARCHITECTURES IN SELECTED OTHER PROBLEMS.

Classes	accuracy	compact?
$(1, 2)(3, 4)(5, 6)$	99.1%	YES
$(1)(2)(3, 4, 5, 6)$	99.5%	YES
$(1, 4)(2, 5)(3, 6)$	56.1%	NO
$(2, 4)(1, 3, 5, 6)$	74.7%	NO
$(3, 5)(1, 2, 4, 6)$	68.3%	NO
$(4, 5)(1, 2, 5, 6)$	74.1%	NO

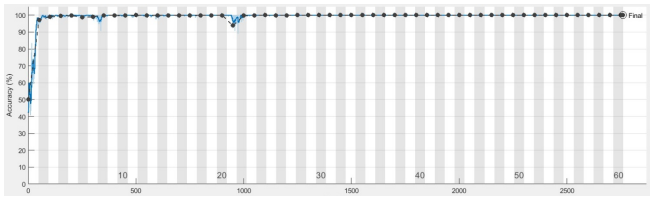

Fig. 9. Training (continuous line) and validation (circles) accuracy plots for CNN1 learning numbers divisible by 3.

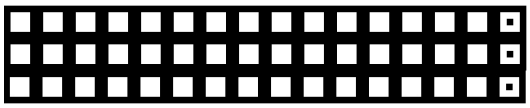

Fig. 10. First-layer of 3×3 filters for (from top to bottom): CNN1 for learning 1 to 6 numbers, CNN2 for learning *even* and *odd* numbers and CNN1 for learning numbers divisible by 3.

dataset, and the learning curve is very steep (see Fig. 9).

Very similar results are obtained for another (somehow artificial) case, where the first class consists of $(2, 6)$ and all other numbers form the second class.

D. Explainability issues

The complexity of proposed architectures is (deliberately) very low to correspond to low complexity of the input data. Therefore, it is possible to inspect the learned values of NN parameters, and informally explain their roles in the process.

Again, the conclusions are very different for FcNN and CNN architectures. In FcNN structures, it is hardly possible to identify any intuitive explanations for the obtained weights. The numbers look random, and even if some specific features (e.g. near-zero columns in the weight matrix of the second hidden layer) can be noticed, they do not exist in other nets trained to learn similar concepts.

For CNN structures, however, the convolutional layers of successfully trained architectures are almost identical. Fig. 10 shows visual representations of 3×3 filters of the first convolutional layer for several such cases. It looks obvious that in all of them only one filter plays an active role (others are approx. averaging filters) and its apparent role is to detect isolated pixels.

Fig. 11 presents the corresponding filters for an unsuccessfully trained CNN1 (see Fig. 5) for learning *even* and *odd* numbers. Again, only one active filter can be noticed, but its effective functionality is unclear. Thus, the failure of this architecture can be attributed to unsuccessful building of the first-layer filters.

Similarly, 2×2 filters of the second convolutional layer are virtually identical for all CNN architectures, and (again) only

Fig. 11. First-layer of 3×3 filters in CNN1 unsuccessfully trained to distiguish between *even* and *odd* numbers.

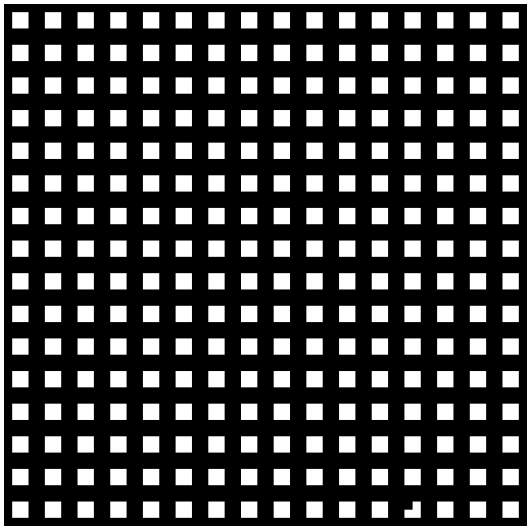

Fig. 12. Second-layer of 2×2 filters for CNN architectures. They are virtually the same for all trained CNN1 and CNN2 structures. Only one filter (in the last row) seems to perform actively.

one of the filters seems active (see Fig. 12). Apparently, its role is to accumulate large-magnitude instances of the first-layer results, i.e. to effectively count the number of dots.

Thus, the inferred explanations accurately correspond to the common-sense understanding of the investigated learning tasks.

IV. Summary

The paper presents some insights into processes of learning basic numerical concepts from visual data (small near-binary images containing several single-pixel dots) by using simple FcNN and CNN architectures.

It was found that in case of concepts with classes forming *compact* subsets in the domain of integers (e.g. (1), (2) and $(3, 4, 5, 6)$) both FcNN's and CNN's can be trained to achieve nearly perfect accuracy on test datasets. However, for *distributed* classes (e.g. *even* and *odd* numbers) the conclusions are less straightforward.

For *distributed* classes, FcNN architectures are sometimes able to achieve accuracies above the random-choice level, but generally their performances are unsatisfactory. Typically, CNN's in CNN1 variant also do not go above the random-choice levels. However, CNN2 architectures (where learning numbers is embedded into a hidden layer of the net) can achieve very high (or almost perfect) accuracies even for *distributed* classes. There are, nevertheless, specific cases where both CNN2 and CNN1 architectures can achieve almost perfect accuracy after very short learning period (with very steep learning curve). Further experiments and analysis are needed to better understand this phenomenon.

In many aspects, the obtained results supplement conclusions from past works on similar topics (mainly [6], [7] and [8]). In particular, we can (partially) support opinions from [7] about limited usefulness of fully connected architectures for counting tasks in visual data. The concepts of using small-scale CNN's for learning numerical abstractions (proposed in [6]) is also confirmed.

However, conclusions from our earlier work [8] about limited capabilities of CNN architectures should be significantly revised.

Additionally, we found that in the investigated problems CNN architectures can be much better explained. In particular, the weights of convolutional filters nicely correspond to the intuitive understanding of the problems. In FcNN architectures, we did not find any regularities coincidenting with the nature of learned problems.

Last but not least, our investigations can be (distantly) related to works discussing learning numerical concepts from neuropsychological perspectives. For example, relations between numerical and logical quantifiers are discussed in [5], while early stages of sensory-based counting abilities and understanding numbers by animals (from insects to humans) are presented in [1], [10], [11], [12] (and many other works). We believe that similar investigations can be continued in the domain of AI agents and systems.

References

[1] A. J. Kersey and J. F. Cantlon, "Primitive concepts of number and the developing human brain," *Language Learning and Development*, vol. 13, no. 2, pp. 191–214, 2017. doi: 10.1080/15475441.2016.1264878

[2] M. H. Fischer and S. Shaki, "Number concepts: abstract and embodied," *Phil. Trans. Royal Society B*, vol. 373, no. 1752, p. 20170125, 2018. doi: 10.1098/rstb.2017.0125

[3] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *Proceedings of the 14th European Conference on Computer Vision, part II*, vol. LNCS 9906, 2016. doi: 10.1007/978-3-319-46475-6_41 pp. 660–676.

[4] S. Sabathiel, J. L. McClelland, and T. Solstad, "Emerging representations for counting in a neural network agent interacting with a multimodal environment," vol. ALIFE 2020: The 2020 Conference on Artificial Life, 2020. doi: 10.1162/isal_a_00333 pp. 736–743.

[5] V. Troiani, J. E. Peelle, R. Clark, and M. Grossman, "Is it logical to count on quantifiers? dissociable neural networks underlying numerical and logical quantifiers," *Neuropsychologia*, vol. 47, no. 1, pp. 104–111, 2009. doi: https://doi.org/10.1016/j.neuropsychologia.2008.08.015

[6] C. Creatore, S. Sabathiel, and T. Solstad, "Learning exact enumeration and approximate estimation in deep neural network models," *Cognition*, vol. 215, p. 104815, 2021. doi: 10.1016/j.cognition.2021.104815

[7] S. Guan and M. Loew, "Understanding the ability of deep neural networks to count connected components in images," in *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2020. doi: 10.1109/AIPR50011.2020.9425331 pp. 1–7.

[8] A. Śluzek, "Counting dots: On learning numerical concepts from visual data," *Proceedings of the 3rd Polish Conference on Artificial Intelligence, April 2022, Gdynia, Poland*, pp. 16–19, 2022.

[9] M. Fang, Z. Zhou, S. Chen, and J. L. McClelland, "Can a recurrent neural network learn to count things?" *Cognitive Science*, 2018.

[10] A. Cope, E. Vasilaki, D. Minors, C. Sabo, J. Marshall, and A. Barron, "Abstract concept learning in a simple neural network inspired by the insect brain," *PLoS Computational Biology*, vol. 14, no. 9, p. e1006435, 2018. doi: 10.1371/journal.pcbi.1006435

[11] M. Tomonaga and T. Matsuzawa, "Enumeration of briefly presented items by the chimpanzee (pan troglodytes) and humans (homo sapiens)," *Animal Learning & Behavior*, vol. 30, p. 143–157, 2002. doi: 10.3758/BF03192916

[12] K. Wynn, "Children's understanding of counting," *Cognition*, vol. 36, no. 2, pp. 155–193, 1990. doi: 10.1016/0010-0277(90)90003-3